Technical Disclosure Commons

Defensive Publications Series

October 02, 2019

Testing Code Using Synthetic Data

Dietmar Dorr

Abhayjeet Singh

Testing Code Using Synthetic Data

ABSTRACT

Testing of software, e.g., regression testing, is often performed using test or mock accounts, rather than with production data. Use of data from real accounts can provide better testing compared with data from test or mock accounts. However, the use of production data is often not possible, e.g., due to risks of the exposure of personally identifiable information (PII). This disclosure describes techniques that enable software testing to be conducted with production data in a privacy-preserving manner.

KEYWORDS

- Regression testing
- Software testing
- Synthetic data
- Production testing
- Scalable software testing

BACKGROUND

Regression testing is a procedure to compare the outputs of two versions of a codebase to ensure that they are consistent, e.g., to ensure that the more recent version has not adversely affected existing features. Regression testing is carried out for different types of software such as operating systems, browsers, other apps, etc.

Testing of software, e.g., regression testing, is often performed using test or mock accounts, rather than with production data. Use of data from real accounts can provide better testing compared with data from test or mock accounts. However, the use of production data is often not possible, e.g., due to risks of the exposure of personally identifiable information (PII).

Therefore, regression testing has traditionally been carried out using test or mock accounts. It is generally difficult to generate representative test accounts due to the diversity in real accounts, which can number in the billions for popular software or software services. Test accounts also elicit fake server responses and don't result in high-quality, representative, performance data. The use of test accounts in regression testing can lead to low confidence in the rolled-out software.

To predict the impact of a change in code reliably, regression testing is also optimally done across a relatively large number of production accounts that represent a substantial share of the overall user base, e.g., millions of accounts, with concise and informative diff-reports generated across software versions. Thus, scalability is also an attribute of good regression testing.

The current development cycle, e.g., the time taken to roll out a code change, can be six-to-ten weeks long, which can be an unacceptably high figure for many software products and/or services. This is because, as illustrated in Fig. 1 below, the development cycle, lacking access to production data, can include several rounds of testing with progressively larger groups of testers (102-8). No reliable way is currently available to prototype small changes to the code quickly. At the same time, the cost of rolling out erroneous code to the user base is very high.

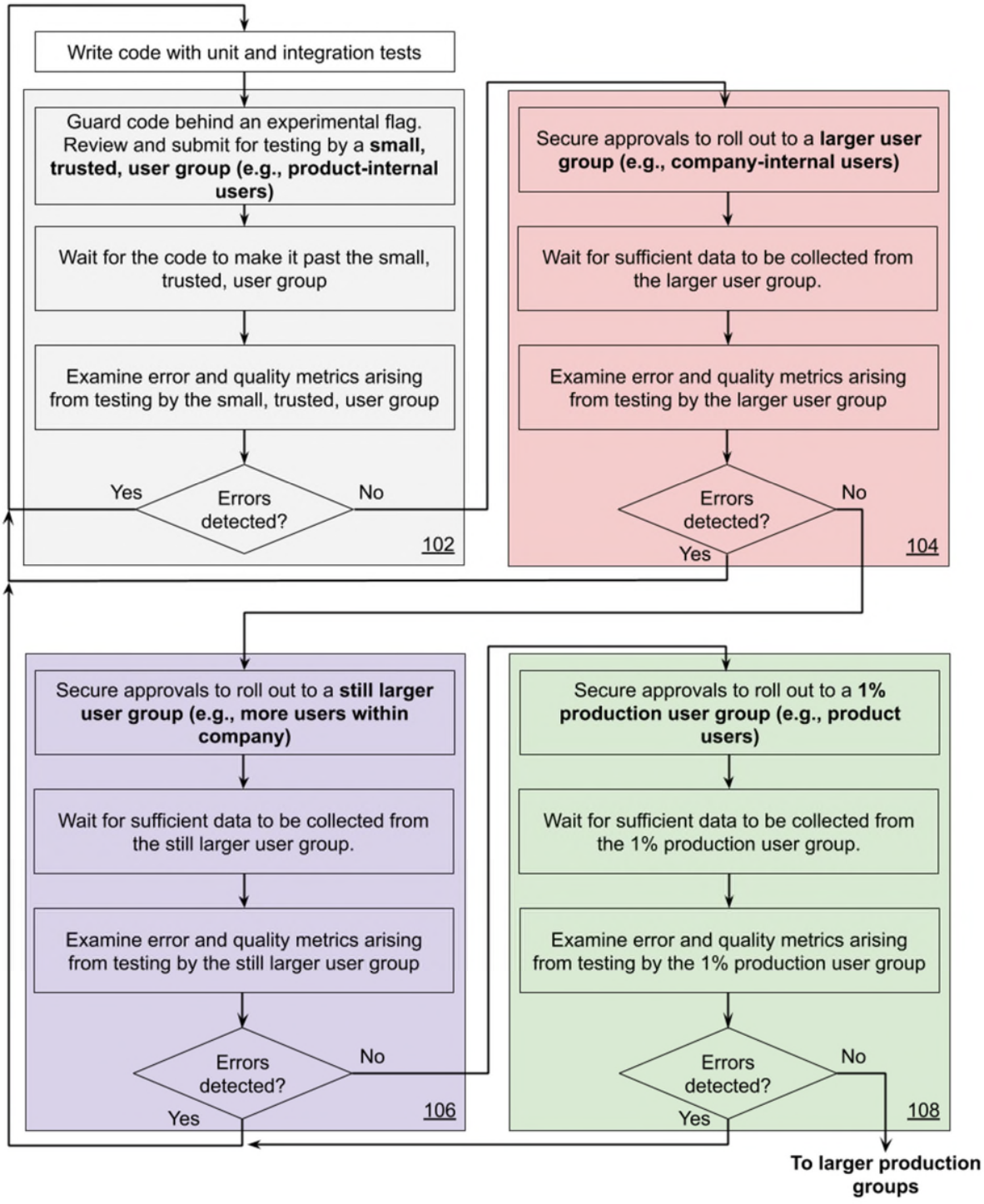

Fig. 1: Example development and test cycle

DESCRIPTION

This disclosure describes techniques to run large-scale regression tests against production data in a secure and privacy-preserving manner.

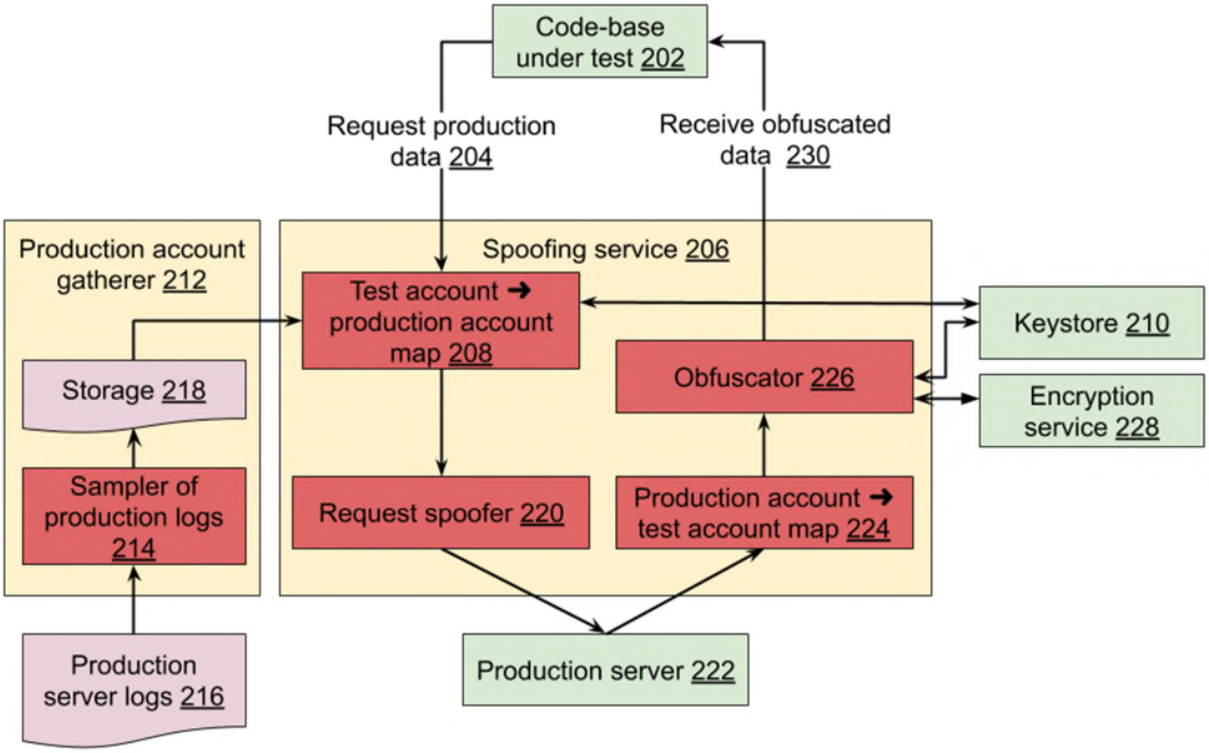

Fig. 2: Testing a codebase using encrypted or obfuscated production data

Fig. 2 illustrates the testing of a codebase using encrypted or obfuscated production data, per the techniques of this disclosure. Production data is used with user permission; users are provided with information about the possible use of their data for testing and can choose to grant or deny permission for testing. A codebase under test (202) makes a request to access production data, e.g., data from production accounts. A spoofing service (or spoof server, 206) encrypts or obfuscates production data, such that the codebase under test can be tested on real data without exposing such data. To use a cryptographic paradigm, the spoofing service is similar to a man-in-

the-middle (MITM). The spoofing service feeds the differencing tool of the regression tester, as explained in detail later.

The spoofing service returns encrypted or obfuscated production data using the following procedure. A test account is mapped to a production account (208). The map is performed by using a hash function to compute an index for a given test account and by using the index to pick a production account from a pool of available production accounts. The hash function is derived from a keystore (210) and can change on a periodic, e.g., daily, basis.

The pool of production accounts is provided to the spoofing service by a production account gatherer (212). The production account gatherer derives the pool of production accounts by running a sampler (214) on production server logs (216) and by storing the pool on storage (218) that is accessible by the spoofing service.

The codebase request to access production data may contain references to test accounts; therefore, the request itself is spoofed by a request spoofer (220). The request spoofer replaces references to test accounts within the request by production accounts. The production server (222), which can be queried using remote procedure call (RPC), responds to the request by providing data pertaining to the production account to the spoofing service. The spoofing service maps the data pertaining to a production account to a test account (224). An obfuscator (226) obfuscates, e.g., encrypts, redacts, or otherwise modifies, production account data in a manner that is reversible, reproducible, consistent, format-preserving, and largely length-preserving. The structure of the production data is thus utilized to generate synthetic data. Some examples of synthetic data generated based on production data are illustrated in Table 1 below.

Data Type	Original Value	Obfuscated Value
Name	John Example	Fjdilux dsfieols
Phone Number	1112223333	73X2XX1TT4
Phone Number	(111)222-3333	(73X)2XX-1TT4
Email Address	sample@example.com	iekse@dsfieols.hudj
Identity Number	876335438990	2564354687980

Table 1: Example obfuscations

The obfuscator communicates with the keystore and an encryption service (228) to execute the obfuscation. The obfuscated data is received (230) by the codebase under test. The codebase is tested against the obfuscated data.

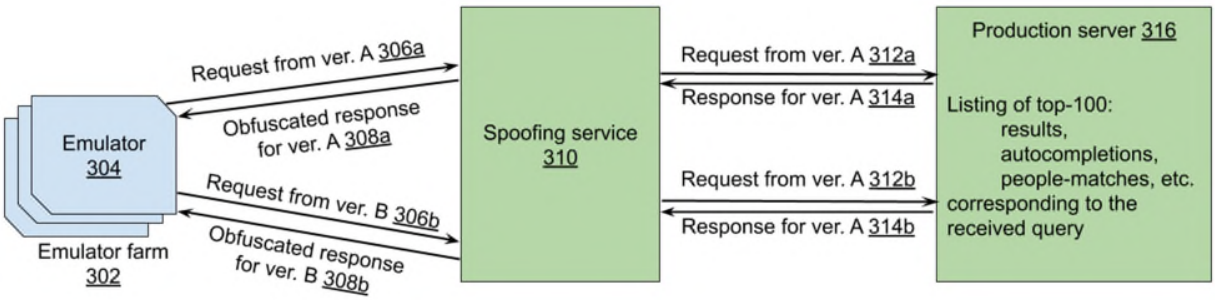

Fig. 3: The use of a spoofing service in the context of regression testing

Fig. 3 illustrates the use of a spoofing service in the context of regression testing, per techniques of this disclosure. One or more emulators (304), which can be part of an emulator farm (302), run different versions of a codebase under test. The versions of the codebase under test are illustratively labeled as A and B. Codebase versions A and B, running on the emulators, send to the spoofing service requests for production data (306a, 305b). The spoofing service (310) intercepts and processes these requests in the manner explained before, and transmits the

processed requests (312a, 312b) to the production server (316). The production server responds to the requests in the form of responses (314a, 314b), which can include, for example, a listing of the top results, auto-completions, matches, etc., corresponding to the received query. The spoofing service receives the production-server responses and obfuscates or encrypts them. The spoofing service transmits the obfuscated or encrypted responses (308a,b) to the emulators, which test the codebase versions A and B using the obfuscated production-server responses. A difference report that highlights performance and quality differences between the outputs of codebase versions A and B is generated.

The techniques described herein leverage the observation that for difference-testing, actual data from production accounts is not needed; rather, it is sufficient to know that data being used for the test is different from the true data, with the data used for the test being gibberish.

An environment that enables testing code using encrypted production data, per techniques of this disclosure, includes the following:

- A hardware/software setup, e.g., an emulator farm to host the codebase under test.
- A test suite, e.g., a set of tests to run on the codebase versions.
- A spoof server or service, e.g., a server, as described herein, that intercepts emulator requests and modifies production data
- A differencing system, e.g., a component that differences the responses (outputs) of the versions of the codebase and provides a user interface for viewing the differences.

CONCLUSION

This disclosure describes techniques that enable software testing to be conducted with production data in a secure and privacy-preserving manner.